MVR Horizontal Tower Defense Kit

Game documentation and HowTo guide.

This document contains:

Package Description and features

Martians VS Robots is a starter kit and example project for Horizontal Tower Defense games. The main objective of the game is to build defensive units on a grid, while enemy units try to breach through your defenses.

This project is a collaboration between Puppeteer & Jordan Swapp

Available in JS and C#

How to Play?

At the start of each level you may choose several units to be used in battle, along with the essential Extractor unit which produces crystals. Collect crystals to build defensive units, and prevent the enemy from reaching your colony.

To build a unit, click on it and then click on an available tile to place it. You can also sell units by clicking on the sell button and then clicking on a unit. Hold H to show the health of all units and enemies.

Showcase and tutorial videos

The following videos showcase gameplay, as well as process guides on how to create unique units and enemies. More videos will be added regularly.

MVR starter kit gameplay
MVR – Halloween gameplay
MVR – Xmas gameplay
MVR – Making a new level
MVR – making a new enemy
MVR – making a new unit

Features

MVR Horizontal Tower Defense provides a working example for developers interested in making games similar to some of the top titles in the market, while adding features to improve functionality and allow you to customize your creation:

- Entire example project included, complemented by documentation, a component details sheet and video tutorials.
- Available in both C# and JavaScript, in the same package.

- Cute 2D quad based graphics, all packed into a texture to reduce drawcalls. A tutorial explains how to add your own graphics using Flash, 3DS Max, and Unity.
- New Xmas theme utilizes the built in Unity 4.3 sprite based graphics, for better performance, nicer animations, and a much easier work process.
- Using Build, Weapon, Enemy and Shot components (or a combination of those) you can create unique and diverse behaviors. All units and enemies in the game are made with those components, Check out video tutorials to see how we create a unit that shoots homing missiles, and an enemy with a turret for a head!
- BuildController and UnitSelection components allow you to setup a build list of units, and let the player choose which units to take into battle.
- EnemyDispencer component allows you to setup waves of enemies to attack you, as well as a level-end boss.
- The LevelSelection component lets you display your levels in a grid, along with lock/unlock status recorded in PlayerPrefs.
- Components such as Sticky (latches unto objects for damage), Gib (make parts fly off as a unit loses health), and ChangeAttribute(change the enemy behavior as it loses health) enhance gameplay with interesting behaviors.
- End of level 1-6 star rating based on your performance. Star rating is recorded in the PlayerPrefs.

Current version 1.2

Update history

(1.2) 19.03.2015
- You can now create buildings larger than 1x1 tiles, such as 2x2, 2x1, 1x4, etc (ex: You can make a building similar to the Cob Cannon from Plant vs Zombies which sits on 2x1 tiles)
- Improved support for gamepads; you can select a unity from the menu and then place it on the ground more easily. Custom pointer is no longer needed.
- Added an option to set the level name prefix. Level status is now loaded correctly.

(1.15) 22.02.2015
- Added support for Unity 4.6 UI, while maintaining the ability to use the old GUI as well.
- As a result, the game now has better support for different mobile devices and screens.
- The game also supports gamepad controls, with on-screen cursor for gamepads.

(1.09) 25.01.2013
- Added a new money system in which you can assign more than one type of money for the player. Accordingly, you can set multiple costs for each unit individually.
- You can set a list of possible items that are dropped from an enemy when it dies. Each item has a separate drop chance.

(1.06) 10.12.2013

- Added Xmas theme with all new units, enemies, and effects. This theme utilizes the Unity 4.3 sprite based graphics as well as the built-in animation system, allowing for nicer animations, better performance, and a simpler creation process.
- You can now set the animations for Idle/Walk/Attack/Die from the inspector. If no animation is set in the inspector, it defaults to the original "Idle"/"Walk"/"Attack"/"Die" animation labels.

- You can set the timing of attacks for weapons and enemies. This allows for better synchronized attack animations. (ex: in the elf animation the elf first aims and then throws the snowball. The snowball is actually created when the animation reaches the correct time instead of being thrown immediately).

- Push back is corrected. It now pushes the enemy in the direction it was shot from. More units in the new theme use pushback to good effect (ex: the bellower).

- Special animation replaces walk animation after attribute change (example: When the gremlin with the shield starts running, its default walk animation becomes the running animation).

(1.0) 28.10.2013

- The entire project has been ported to C# thanks to Jordan Swapp.
- Added an in-game pause menu, and changed the stress scene.
- Added Halloween theme assets with example scene, including new units, enemies, and effects.

(0.93) 23.9.2013

- Added an end of level star rating, which is also registered in PlayerPrefs and displayed in the level selection screen.
- Fixed a case in which cooldowns are counted twice.
- Made the startMenu script more generic, allowing users to add any number of menu items which can run any function on any object by name.

Credits

The sounds are courtesy of the free sound project.

Credits go to these authors for their great sound samples: **sarson, HerbertBoland, WillHiccups, Connum, luffy, juskiddink, ggctuk, jobro, isaac200000, tallers, ecfike, sergenious, scuola-pereto, pushtobreak, themfish, qubodup, lagomen, craiggroshek, scuola-pereto, RHumphries, bennychico11, a various others**

Music track is an edited clip from **Five Armies by Incompetech**

Halloween/Xmas music is a clip from **Moonlight Hall by Incompetech**

Please rate my file, I'd appreciate it 

Overview of the game's library contents

Let's take a look inside the game files. Open the main MVR assets folder using Unity3D 4.6 or newer. Take a look at the project library, usually placed on the right or bottom side of the screen. Here are the various folders inside:

- **Animations:** Holds the animation clips made with Unity's built-in animation system. The Xmas theme uses these animation clips.

- **Fonts:** Holds the fonts used in the game as well as the main GUISkin used. The fonts are ARIAL16, and ARIAL 24.

- **Materials:** Holds all materials used in the game. The units and enemies all use a single material (AllUnits_512x512), while the land object has another material (Land). Particle effects also have their own materials, which use the same texture with tiling and offset.

- **Models:** Holds all FBX models imported over from 3DS Max. The "3D" models are just simple quads with a texture and UV mapping assigned to them. Some more complex models contain several quads, each holding a different part, and all linked to each other and sometimes animated.

- **Prefabs:** Holds all the prefabs used in the game. These are distributed to various folders for easier access, including Units, Enemies, Effects, etc

- **Scenes:** The first scene that runs in the game is StartMenu. From this scene you can get to any of the 12 levels. The naming convention for levels in this package is "LEVEL" + number. The last scene is Victory which is a simple message displayed when you win. There is also a StressTest scene to test out performance.

- **Scripts:** Holds all the scripts used in the game. Each prefab contains one or more of these scripts. To learn more about the editable variables in the scripts, check out the component sheet.

- **Sounds:** Holds all the sounds used in the game. Explosions, UI effects, attacks and deaths, etc

- **Textures:** Holds all the textures used in the game including units, background, textures for the various UI elements such as buttons, titles, buttons, etc.

Component details sheet

The following is a table of all the components in the package, their variables, and some details about them:

CLICK HERE TO OPEN COMPONENT DETAILS SHEET IN EXCEL

Customization guides and videos

Getting started

MVR is considered a complete project, and as such is supposed to work as the starting point of your planned game, rather than an addition to an existing project. That said, you may of course pick and choose some of the scripts/models to import into you existing project, but MVR works best as a starter kit which you can customize any part of to your liking.

In the following videos we will show customization processes that cover a wide range of topics using several tools (3DS max, Flash, Unity3D). We will change the land graphics and add a new level, and give the new level a unique set of enemies and unit selection.

We will also see how to create an entirely new unit right from the drawing phase in Flash, through the hierarchy in 3DS Max, and finally into Unity where we will assign it with unique values and add it to the list of available units.

The third video will show us how to create an enemy using some of the graphics already available, and then give it unique behaviour and add it to the enemy list.

Creating a new level

We will create a new level with different graphics and a 5x8 layout. The process requires the following tools: Flash CS3+ and Unity 4.

We will create a land texture in Flash and then export it as PNG. The texture is assigned to a plane of the same size (1024x1024) in 3DS Max, but that is not needed here as we already have a ready FBX model for the land in Unity.

In Unity we will add several prefabs and components such as the GameController, EnemyDispenser, MoneyMaker, DefeatLine, and others. We will configure those components to make a unique level.

Finally, we will test it out and add it to the list of levels in the game as LEVEL 13.

CLICK HERE TO WATCH VIDEO

Creating a new unit

In the following video will create a new unit with unique functionality: A sam site that launches homing missiles at enemies.

The process requires the following tools: Flash CS3+, 3DS Max, and Unity 4.

In this package all the units and enemies are drawn on a single texture in order to conserve drawcalls. The texture is drawn using Flash, and is then exported as a PNG to 3DS Max. The PNG is assigned to a plane of the same size (512x512), and then each unit is "sliced" out to its own quad.
When all parts of a unit are arranged together they are exported as FBX to Unity. Finally, we add all the required components to the unit.

CLICK HERE TO WATCH VIDEO

Creating a new enemy

We will create a new enemy that can also shoot at the player's units.
The process requires the following tools: 3DS Max and Unity 4.

This time we'll skip the Flash portion and reuse some of the parts from other enemies. In 3DS Max we will use the animated legs of a Charger, along with the head of a Turret. In Unity we will setup the Enemy component, and add a Weapon component to make the turret shoot

Finally, we will test it out and add it to the list of Enemies in LEVEL 1.

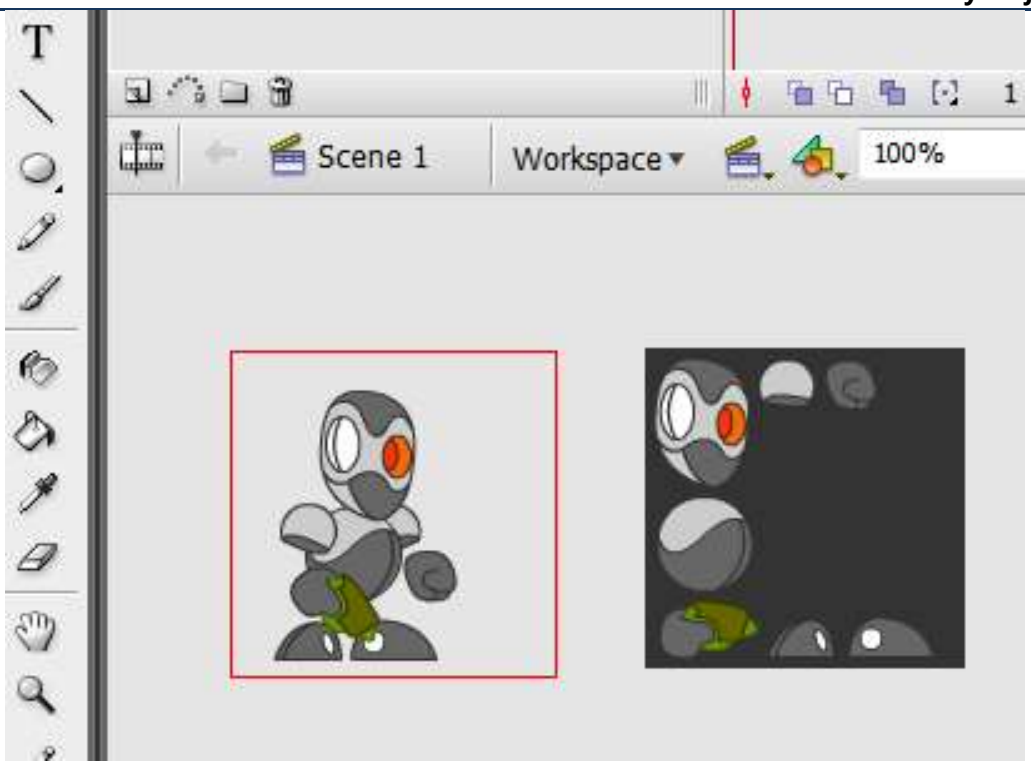
CLICK HERE TO WATCH VIDEO

Creating an animated unit with Unity 4.3

Since Unity introduced the new sprite system, it has become much easier to import graphics into your game. Coupled with the built-in animation system, the entire process of drawing, importing, and animating units and enemies has become easier than ever.

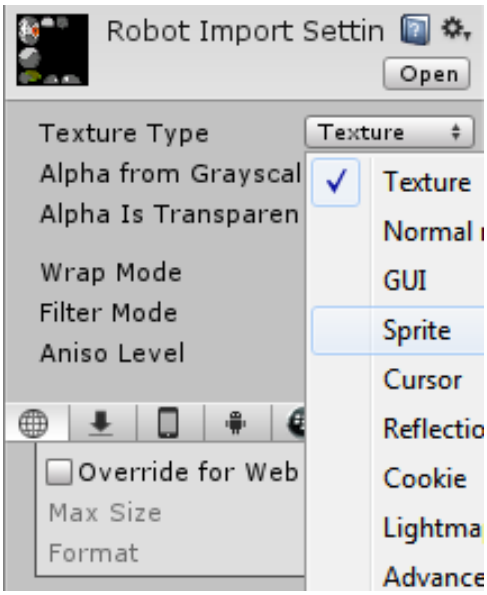
In this section we will create a simple character in Flash. We will then export it as PNG. In Unity we will slice the PNG into useable sprites. After that we will duplicate one of the available units, change its graphics into our own, and create a new animation for it. Let's start!

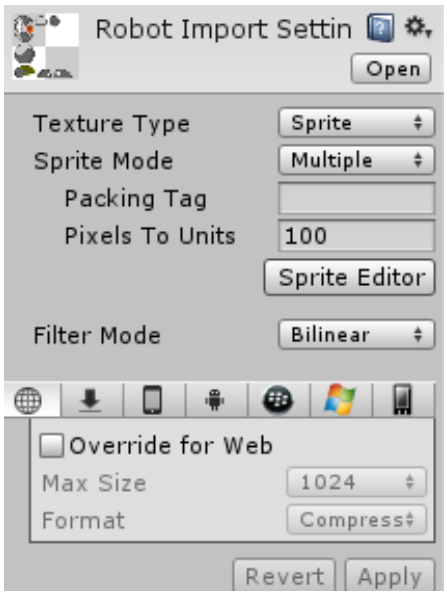
1. Open Flash or your favorite illustration software and start painting. I made a simple robot character, making sure it fit within my default tile size for the game (128x128). After finishing the drawing I arranged the different parts of the robot into a 128x128 canvas. Notice that I didn't copy over all parts of the character as they are redundant, such as the shoulder piece.



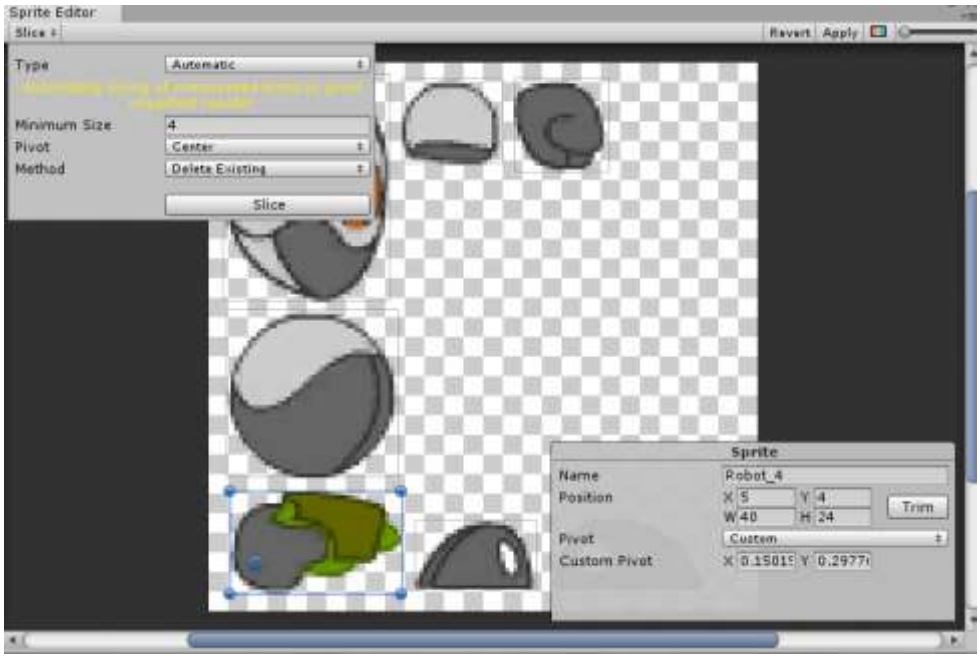
2. After we are satisfied with the result we export the image into PNG. Now moving on to Unity.

3. In Unity, select your PNG texture in the inspector and change the texture type to Sprite, then change Sprite Mode to Multiple.



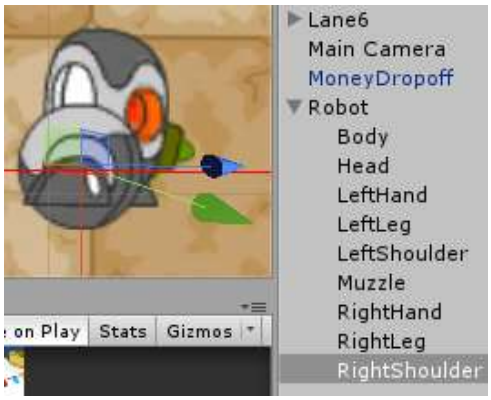


4. Click on Sprite Editor so we can slice up the sprites. Click on slice and choose automatic, then slice. This creates bounding boxes around each part of the graphics. Here you can also change the pivot of the sprites for use in animation.

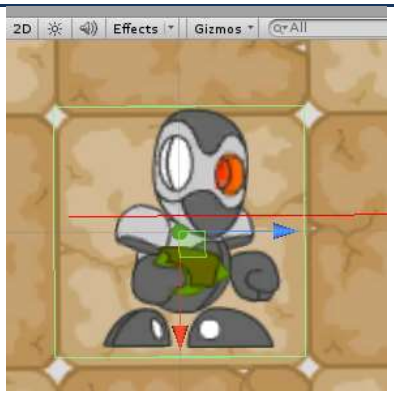


5. Now that we have our sprites ready, let's use an existing unit as a template for our robot warrior. Choose the Elf from the Xmas theme Units folder, duplicate it and rename it to Robot. Drag the Robot prefab to the stage so we can edit it more easily. Make sure to place the prefab on a position that aligns with the game tiles. A good way to make sure your prefab is aligned correctly is to select it while Gizmos button is toggled on. This way you can see the collider gizmo of the prefab and place it right on the tile. My project is setup so that each tile corresponds to a coordinate, so changing the prefab's position to 0,0,0 can help you move it along the tiles while holding Ctrl.

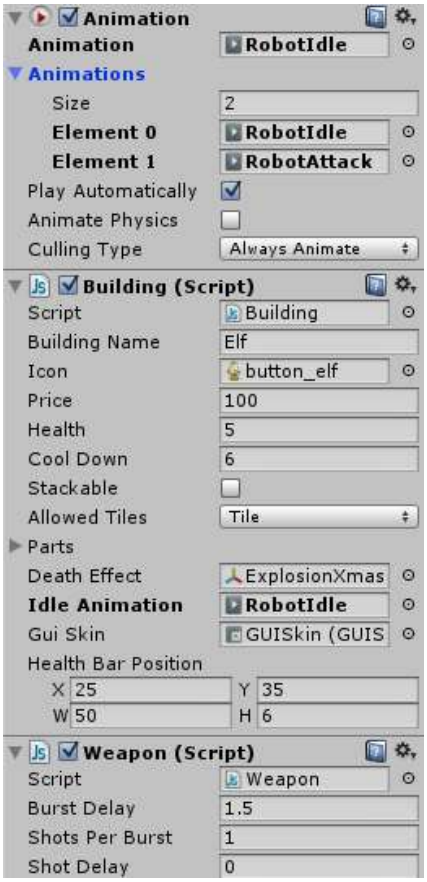
6. Look in the hierarchy of the Robot and you'll see all the parts of the Elf we duplicated before. Delete all the parts except one part, as well as the object named Muzzle. Select the part, rename it to Head and change the sprite to the sprite of the robot head. Duplicate the Head and rename it to LeftLeg, then assign the correct sprite. Repeat for all the remaining parts (Head, LeftLeg, RightLeg, LeftHand, RightHand, LeftShoulder, RightShoulder, Body).



7. Now we must arrange the sprite to resemble our intended robot character. Also make sure to order them in the Y axis correctly (shoulder behind body, left leg behind right leg, etc).



8. Duplicate one of the existing animations twice (we duplicate them instead of creating new ones because we want to keep their Legacy status, which doesn't exist in a newly created animation) and rename them to: RobotIdle, and RobotAttack, and assign them to the Robot's animation component. Also assign RobotIdle to the Idle Animation field in the Building component, and assign RobotAttack to the Attack Animation field in the Weapon component.



9. While selecting the Robot object press Ctrl+6 to enter animation mode. If you're familiar with the basics of animation you'll feel at home here. Move to a point on the timeline, and then rotate or drag

one of the robot parts to animate it. We need to create an idle animation and an attack animation with the robot shooting.

10. Finally select the animations we created, set RobotIdle to WrapMode loop, and RobotAttack WrapMode to Default. We do that because we want the idle animation to keep looping, but we want the attack animation to run once each time it's played.
11. If you check out our Robot in-game now you'll see it shoots at approaching enemies, playing the animations correctly. But it's still not perfect. The robot shoots snowball out from an incorrect muzzle point and the animation timing of the shot is way off. Let's fix those things.
12. Select the robot and replace the ElfShot in the weapon component with a regular shot. Also, move the Muzzle part of the robot to the point at which the gun is about to shoot in the animation. Finally, change the Attack Animation Time to 0.3 which is exactly the frame in which the robot shoots.

That's it, we're done! :D

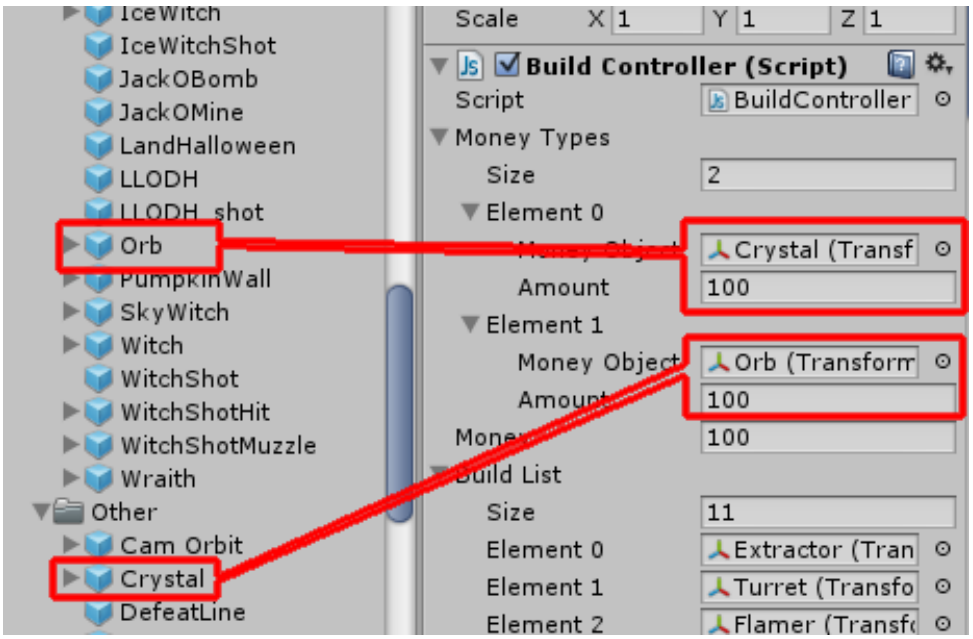
Using multiple money types

Since update 1.09 you can now assign multiple money types to be accumulated and spent during the game. Each money type has a separate amount value.

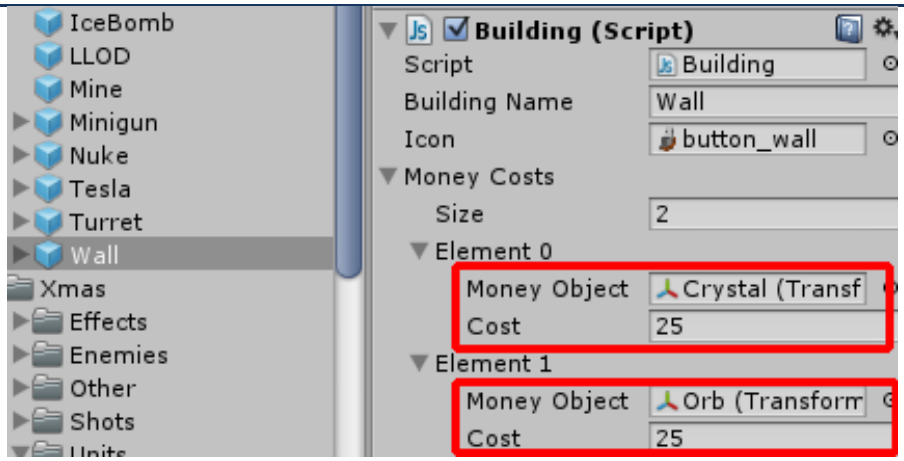
To start using this new system we must assign money types to our GameController (so that we see what money types we have), as well as the units that we wish to build (so that we can see which money types are required to buy a unit). Once both values are set, this new system will override the old single money type system.

Note: You can still use the old single money type system by simply keeping the moneyTypes array at 0 length. This will make the game default to the single money type from before update 1.09.

Take a look at the image below to see how we assigned the money types to our level's GameController. We increased the moneyTypes array length to 2 and then dragged the Crystal and Orb money objects to their respective slots. After that we set the amount of each money type.



Similarly, we can set the money costs of a unit by dragging the money types to the Building component and setting their amounts. In the example below we set the Wall building so that it requires 25 Crystal and 25 Orb in order to be built. You don't need to set both types for a unit. For example we could just assign a Crystal object in the Building component which means the unit needs only Crystal to be built. Look at all the other units to see their money types costs.

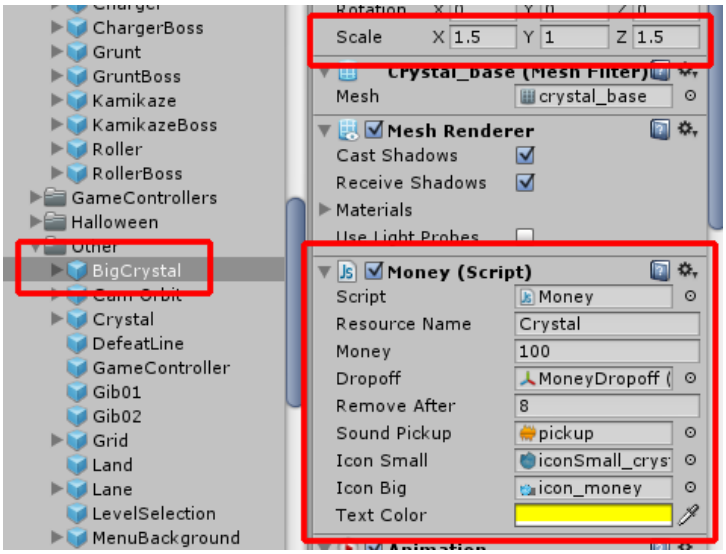


Creating a new money type

You can create your own money type and assign icons and value on pickup. Money is picked up when the mouse moves over it. The game knows which type of money we picked up by comparing its name to the money types the player has.

Let's try it out by creating a new BigCrystal money type which will be a larger amount variation of the basic Crystal money type. We won't need to add a new money type to the GameController because it will have the same name as Crystal, just a different value and in-game graphic. We will then assign it to the MoneyMaker of the lad object so that it is created periodically in the game area.

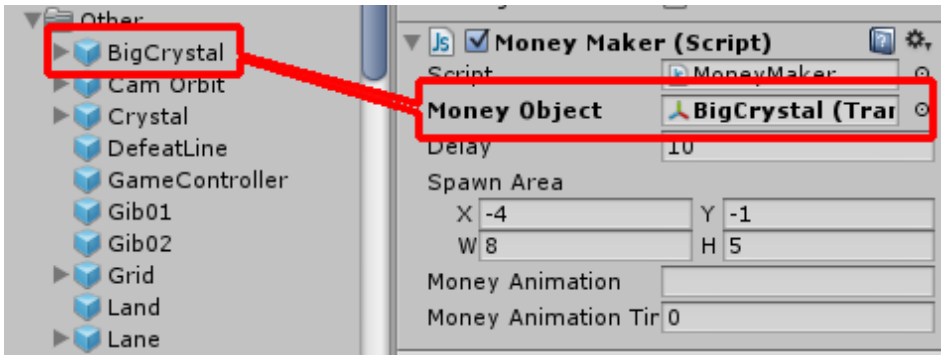
First off, duplicate the existing Crystal money type by selecting it and pressing Ctrl+D. Now let's look at what we have in the BigCrystal Money component.



1. First we increase the scale of this money object to 1.5 so it can be differentiated from the small variation.

2. We keep Resource Name the same, because this is still a Crystal type and it should be added to the amount of Crystal we collected.
3. We increase the money amount to 100. When picked up, the amount of Crystal we have will be increased by 100.
4. The Dropoff object is the same for all money types. You can see it in the game area in the bottom left side.
5. We set it so that the money disappears after 8 seconds if not picked up.
6. We set the sound clip when picking up this money object.
7. We set the small icon of the money which is displayed on the unit card on the top of the screen.
8. We set the big icon of the money which is displayed in the list of money types we have on the bottom of the screen.
9. We set the color of the amount displayed for this money type to yellow.

Finally, in order to test it out we select the land object and drag BigCrystal to the MoneyMake component.



Now when you play this level you'll see that every 10 seconds a big crystal is created. Pick it up and it adds 100 to your Crystal amount! Well done!

Frequently Asked Questions

Does this package work on mobile?

Yes, this package has been successfully tested on both Android and iOS devices. The mobile controls are based on the default mouse controls, meaning that a click on PC equals a tap on the mobile device.

My sprites are not showing on iOS

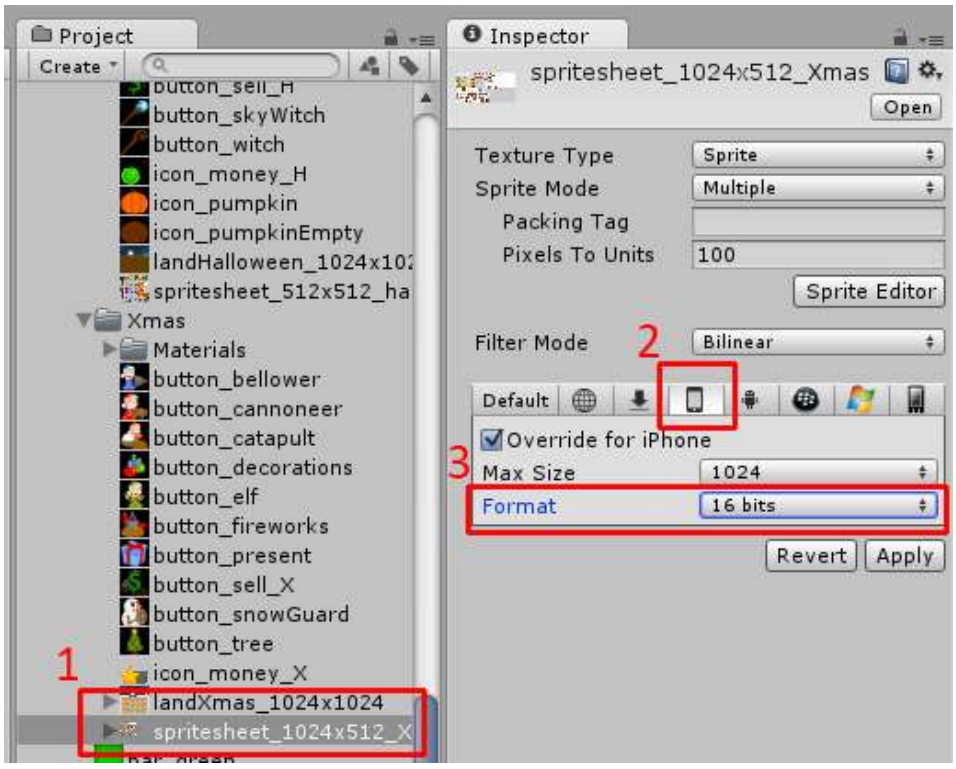
Sprite-based textures made with the new Unity 4.3 can sometimes disappear when working on the iOS platform.

You can notice this by opening a scene and placing any unit made with the Xmas theme in the scene. When you switch from your current platform to the iOS platform the sprite textures become invisible.

To solve this we must change the texture compression format for iOS. Follow these steps:

1. Click on spritesheet_1024x512_Xmas in the project view.
2. Click on the override for iphone button on the right side.
3. Change the format to 16bit.
4. Click Apply.

Look at this image to show you where to click:



Can I still use the old graphics/animation process?

Yes, you can still follow the video tutorials for creating and animating graphics using 3DS Max quads. The default MvR demo still uses this method, while the new Xmas theme uses the sprites and built-in animation tools. Read here on how to use the new graphics/animation process.

What are the default animation names for Enemies/Weapons?

If no animations are assigned in the components for Enemy/Weapon (drag and drop an animation clip to the required field), the default animation names are played. These are “walk”,”attack”,”die” for Enemy, and “attack” for Weapon.

If you have any other questions please post them in the project thread

http://forum.unity3d.com/threads/200355-RELEASED-MVR-Tower-Defense-Starter-Kit

It is highly advised, whether you are a designer or a developer to look further into the code and customize it to your pleasing. See what can be improved upon or changed to make this file work better and faster. Don't hesitate to send me suggestions and feedback to puppeteerint@gmail.com

Good luck with your modifications!